Whole Genome Shotgun Sequencing Tutorial

Frank Olken
Lawrence Berkeley National Laboratory
Berkeley PGA Course
June 15, 2001

Caveats

- Other people's work
- Mostly Celera work
  - Eugene Myers, Ham Smith, et al.

Problem Statement

- Human (and other mammalian) genomes are approx. 3 billion base pairs long
- Sequencing machines can generate sequences for fragments 500-600 bp long
- How sequence human/mouse genome, 500 bp at a time?

Directed Sequencing

- Generate sequence primer
- Run sequencing reaction from genomic DNA, starting from known primer
- Read sequence (500-600 bp)
- Generate next sequence primer
- Repeat
- Expensive (custom primers), slow (sequential)

Shotgun Sequencing

- Extract DNA
- Fragment DNA
- Clone DNA
- Sequence both ends of clones
  - (500-600 bp each read)
- Assemble
- Finish sequencing (close gaps)

Two approaches

- Hierarchical top-down approach
  - Basic strategy of public Human Genome Project (1988 - 2000)
- Whole Genome Shotgun Sequencing
  - Celera strategy for Drosophila and Human Genome (1999-2000)

Hierarchical Top Down Sequencing Strategy

- Sort chromosomes
- For each chromosome clone large fragments of DNA
- Map clones
- Identify spanning set of clones
- Shotgun sequence each clone
- Finish (close gaps)

Whole Genome Shotgun Sequencing

- Take entire human genome
- Construct 3 different sized clone libraries
- Sequence both ends of each clones
- Assemble entire genome
- Finish (close gaps)

Hierarchical Top-Down Strategy

- Conservative older approach
- Mapping (was) cheaper than sequencing
- Assembly computations are easier
- Incremental effort generates useful partial results.

Whole Genome Sequencing Won

- Sequencing became cheap, more accurate
- Sequence more informative (and reliable) than mapping info
- Simpler protocol, easier to automate
- Double ended sequencing, multiple size libraries helped cope with repetitive DNA and gaps
- Assembly computational became feasible

Ongoing Strategy Debate

- Mouse will proceed with hierarchical approach
- Apparently due to difficulty in creating uniform coverage clone library for WGSS.
- Celera, and many others believe WGSS is preferred approach (faster/cheaper).

Remainder of Talk

- How to actually implement shotgun sequencing
- Based primarily on work of Gene Myers, et al. At Celera Corp.
- Last part will concern parallelization issues

WGSS Biology

- Construct 3 clone libraries
- Multiple insert sizes:
  - 1 Mbp, 50 Kbp, 10 Kbp
- Size selection of inserts via gel electrophoresis (prior to cloning)
- Careful library construction to assure uniform coverage

WGSS Innovations

- Double ended sequencing
- Multiple size clone libraries
- Size selection
- Capillary electrophoresis sequencing machines ==> few lane crossing errors
- Longer reads circumvent Alu's
- More accurate reads

Major innovations

- Double ended reads
- Novel assembly algorithms
- Massive compute facilities for assembly
- Estimate: 20K CPU hours per assembly

Shotgun Sequence Assembly

- Has become routine in smaller organisms
- Difficult for large genomes
- Principal problem = repetititve DNA

Shotgun Sequence Assembly

- Data = sequence overlaps
- Overlaps ==> local order (up to reflection)
- Combine local info to get global order (upto reflection)
- Resolve reflection via physical mapping (FISH, etc.)

Shotgun Sequence Assembly

- Compute the overlap graph
- Compute graph layout (linearization)
- Consensus sequence generation

A Digression on Graph Theory

- Graph representation
  - edge list vs. incidence matrix
- Interval Graphs
  - idealization of overlap graph, characterization
- Overlap Graph
- Probe Clone Incidence Graphs

Graph representation

- Edge List
  - list of edges
  - directed edge = (from vertex, to vertex)
  - good for sparse graphs
- Incidence Matrix
  - A (i,j) = 1 if there is an edge from vertex i to vertex j.
  - good for dense graphs, fast computations

Overlap Graph

- Vertices = sequence reads
- Edges between two vertices if corresponding sequences overlap
  - note that we also have to consider alternate strands = reverse complement sequence

Probe Clone Incidence Graph

- Probes = short unique subsequences
- Clones = sequence reads
- Incidence - probe is contained in clone
- Asymmetric matrix = bipartite graph

Human Chromosomes are Linear

- Not circular
- DNA sequence can be mapped onto an interval onto an interval of the integers
- DNA sequence contains no cycles
- DNA sequence contains no branches

Interval Graphs

- Ideal overlap graph is an interval graph
- Interval graph is graph imputed from set of overlapping intervals on real line
- No cycles
- No holes (all subraphs of size four have a diagonal edge)
- No branches

Consecutive One's Property

- There exists a permutation of the incidence matrix representation of the overlap graph such that all of the 1's for a clone (read) are consecutive (no intervening 0's)

Consecutive One's Property

- Example
  - (1 1 1 1 0 0 0 0 0)
  - (0 1 1 1 1 0 0 0 0)
  - (0 0 1 1 1 1 0 0 0)
  - (0 0 0 1 1 1 1 0 0)

Interval Graph Recognition

- Testing to see if a graph is an interval graph can be done in O(E) time - I.e., time linear in the number of edges on a serial machine
- Booth-Leuker algorithm from 1970's

Coverage Issues

- Shotgun sequencing = random sampling of read sequences
- Goal is 10-12 X coverage
- Very expensive - tens millions of dollars
- Actual human genome was 5-6X coverage
- Higher coverage ==> fewer, smaller gaps

Overlap Graph Construction

Naïve Overlap Detection

- Pairwise comparison of all reads
- $O(n^2)$ compute time, where n= # reads
- n = 6X * 3Bbp / 500 bp/read
- n = 36 Million reads
- n^2 = 14.4 * 10**14

Naïve Overlap Detection (cont.)

- Approx. string matching via Dynamic Programming (Smith Waterman)
- O(m*n), where m,n = string lengths = 500
- O(mn) = 25,000
- Assume 10 instruction in inner loop
- Total CPU time =
  - 5*10**20/1GHz = 5*10**11 cpu seconds

Better Overlap Detection

- Low sequencing error rates implies that
- Overlaps will include many exact matches of short DNA sequences
- Example:
  - 20 bp subsequence is unique in human genome
  - 20 bp * 0.1% error rate
  - 98% exact matches for 20 bp sequences

Linear Overlap Detection

- Shred reads into overlapping k-mers (20 bp)
- Build a hash-table of the k-mers
  - hash function = remainder modulo prime no.
  - Entry = (hash key, k-mer, read ID, offset)
- Shred each read into overlapping k-mers
- Look up each k-mer
- Count k-mer matches for each read pair
- run DP approx string match for high scoring pairs of reads

Linear Overlap Detection (cont.)

- This overlap detection algorithm requires
- O(N) cpu time
- where N = sum of lengths of all reads
- Assume coverage, k, is finite (6X)
- Space is also O(N)

Improvements in Overlap Detection

- Use disjoint k-mers for lookup (or table)
- Use only k-mers from both ends of reads for lookups (Olken)
- Use (random) subset of k-mer values (UMD)
- Space/time complexity still O(N), but smaller constant

Naïve Parallel Overlap Detection

- Partition reads (randomly) among cpu's.
- Build hash table for each partition.
- Broadcast reads to all cpu's.
- Lookup in all partitions in parallel.
- Score number of exact matches.
- Run DP approx string match in parallel.
- Output overlaps

Naïve Parallel Overlap Detection

- Algorithm distributes reads across processors effectively partitioning data.
- Permits one to handle very large datasets.
- However, trivial speedup in cpu time.
- Must search every k-mer against every partition.

Overlap Graph Construction Seen as Join Algorithm

- Construct (k-mer, read ID) tuples
- Join on k-mers
- Group on read ID pairs
- Count
- Run DP approx. string match on high scoring pairs

Join-based Overlap Graph Construction

- Joins of k-mers, grouping, etc. can be done two ways
- Sort-merge based join (UMD, 2001)
- Distributive hash join (Olken, 2001)

Sort-based Join

- Sort input records on join key (k-mers)
- Construct cross product (all pairs) of all records with matching join keys
- Distributive Sort Join
  - Sample join keys
  - Distribute input data among cpu's by join key
  - Sort join in each cpu

Distributive Hash Join

- Hash input records on join keys (k-mers)
- Construct cross product (all pairs) of records with matching join keys
- Distributive Hash Join
  - partition input records among cpu's according to hash(join key).
  - Do Hash Join in each CPU

Complexity of Join-based Overlap Graph Construction

- Distributive Hash Join
  - O (N) total work
  - Linear speed up with number of processors
- Sort Join
  - sort-merge = O(N*log(N)) work

Overlap Graph Layout

Linearization of Overlap Graph

Overlap Graph Layout

- Goal: linear ordering of sequence reads
- Input: overlap graph
  - (plus mated read pair info)
  - unit interval graph is easier

Unit Interval Graph

- Unit interval = interval graph where all intervals are the same size
- Sequence reads are (nearly) the same size
- We are working with unit interval graph
- Implication:
  - no read contained within another read
  - all reads extend to left or right of other reads

Size of input data

- Vertices = 6X * 3Gbp / 500 bp/read
- Vertices = 18 Million
- Edges = 2 * 6X * 18M = 216 M edges
- Assume vertices denoted by 4 byte integer
- Assume edges denoted by 2 x 4 bytes
- Useful to store offsets for overlap edges

Basic Approach

- Bottom up construction
- Use most reliable data first
  - better a partial layout than an erroneous one

Transitive Closure of a Graph

- (a,b) and (b,c) implies (a,c)
- a.k.a. reachability graph
- Add edge (a,c) if there exists a path from a to c
- Assume directed graph
- Used in computing airline ticketing
- Many other applications

Transitive Reduction of a Graph

- Inverse operation to transitive closure
- H=TR(G) is the minimal subgraph of G such that TC(H) contains G
- Assume G is a directed graph
- TC = transitive closure
- TR = transitive reduction
- Remove all edges which can be inferred from remaining edges

Transitive Reduction

- Transitive reduction of unit interval graph = unique order linear graph (in interval order)
- Transitive reduction of an ideal overlap graph gives us the desired graph layout

Computing Transitive Reduction

- TR removes redundant edges
- We can use offset information to facilitate TR computation (by finding cliques)
- Max. path length is bounded due to finite coverage

Alternative approach to reduction of overlap graph

- By Olken, 2001
- Identify & contract cliques in overlap graph
- Cliques = maximal complete subgraphs
  - every vertex is connected to every other vertex in the clique
  - no larger complete subgraph exists which contains the clique

Clique Detection & Contraction

- Identify cliques in overlap graph
  - Construct unique probes from ends of reads
  - Probe the reads (test for containment)
  - Set of reads which contain unique probe forms a clique (maximal complete subgraph) of overlap graph
- Contract cliques
- Reduces graph size by coverage factor

Contraction of unique linear subgraphs

- After transitive reduction we should have a simple ordered list
- Reality = branches exist due to false overlaps
- Contract unique linear subgraphs (unitigs)
- Facilitates subsequent processing of branches

After computing transitive reduction

- We still have to deal with
  - false overlaps
    - from repeats
    - from chimeric clones
  - missing data
    - generates gaps
    - missing overlaps ==> non-interval graph

Branches in Overlap Graph

- May be able to resolve via inconsistent sequence overlap info - overlap stops in middle of read
- Otherwise use Kececioglu's clustering technique to split up sets false overlaps due to repetitive DNA
- If all else fails delete repetitive DNA and use mated pair info to order contigs

Repetitive DNA

- Many kinds, vary in number of repeats, length, degree of similarity, tandem vs. non-tandem
- Alu's = 300 bp, 100K copies
- Lines, Sines - longer, fewer copies
- Gene duplications - (> 1 Kbp), few copies each

Repetitive DNA is a problem

- Generates false overlaps
- Introduces
  - sequence compression (true seq is longer)
  - topological problems in layout graph
  - non-chordal subgraphs (donuts)
  - branches in sequence

Identifying Repetitive DNA

- Common repeats are catalogued
  - e.g., Alu's, Lines, Sines, ….
- Unusually high numbers of overlaps
  - (should be approx K = coverage).
- Anomalous overlaps
  - broken in middle of reads
- Inconsistencies in overlap graph layout -
  - branches, cycles, donuts

Dealing with Repetitive DNA

- Identify putative repeat sequences
- Cluster repetitive seqs on differing positions
- Recompute overlap graph
- Recompute graph layout
- Check for topological errors
- Loop
- See Kececioglu paper at RECOMB 2001

Dealing with Heterozygosity

- Humans have diploid genome
  - 2 copies of chromosomes (not (X,Y) in males)
- Heterozygotic
  - 2 different genes (one from each parent)
  - will not assemble as a single linear chromosome
  - unresolvable branches, donuts
  - find via clustering (Kececioglu)

Scaffold Construction

- Scaffolding = ordering via mated read pairs from ends of clones
- Used to span gaps (repeats, missing data)
- Use smaller clone mates first
- Use multiple mate pairs info first
- Mate info includes est. of gap distance
- Gives adjacency information at contig level

Anchoring to Chromosome Maps

- Final “scaffolding” is mapping onto known chromosome maps
- Anchor via mapped Sequence Tagged Sites
- At least two such anchor points are needed to orient sequence on chromosome

Parallelization of Graph Layout

- Observe that shotgun sequencing generates many disjoint connected components (CC)
- Disjoint connected components can be processed in parallel
- Perform parallel CC labeling
- Move each CC to a single cpu
- Apply serial graph layout to each CC
- by Olken, 2001

Consenus Sequence Generation

Consensus Sequence Generation

- Multiple Sequence Alignment
- Voting = nucleotide estimation for each column in multiple sequence alignment

Multiple Sequence Alignment

- High read accuracy reads - eases MSA
- Optimal algorithm is NP hard
- Common practice = greedy clustering
  - compute pairwise alignments
  - merge most similar pair of sequences (or alignments)
  - update consensus sequence estimate
  - iterate (hierarchical clustering)

Multiple Sequence Alignment

- For tractability and parallelism
- Break apart MSA problem into smaller problems
  - horizontal partitioning at “gaps”
    - gaps either natural or induced

Voting

- Need to estimate nucleotide for each column in the MSA
- Classically use plurality voting in each column
- If we have reliability info for each position from sequence trace analyzer we can be more sophisticated
  - weighted voting, MLE, Bayesian

Finishing

- Shotgun sequencing leaves many small gaps
- Small gaps
  - span via PCR from genomic DNA
  - if contigs are unordered use PCR pooling
- Larger gaps
  - for ordered contigs retrieve spanning clone
  - shotgun or directed sequence the spanning clone

Conclusions

Whole Genome Shotgun Sequencing

- Preferred strategy for large scale sequencing
- Computations are feasible
- Repetitive DNA is the chief difficulty in assembly
- Requires reads from both ends of clones
- Computation can be fully parallelized on distributed memory machines

Acknowledgements

- Funding: U.S. Department of Energy, Office of Biological and Environment Research

Contact Information

- Frank Olken
  - Lawrence Berkeley National Laboratory
  - NERSC Division
  - Mailstop 50B-3238, 1 Cyclotron Road
  - Berkeley, CA 94720
  - Tel: 510-486-5891
  - Email: olken@lbl.gov
  - WWW: http://www.lbl.gov/~olken